

Chapter Goals

- Provide a brief introduction to object-oriented information modeling.
- Provide a brief introduction to directories.
- Provide a brief overview of DEN.
- Show how DEN will be used in Cisco products.

Directory-Enabled Networking

Directory-enabled networking is not a product or even a technology. Rather, it is a *philosophy* that uses the Directory-Enabled Networks (DEN) specification to bind services available in the network to clients using the network. The DEN specification enables applications to leverage the capabilities of the network as well as better support the needs of the applications using it.

DEN is in reality two things:

1. A specification of an object-oriented information model that models network elements and services as part of a managed environment in a repository-independent fashion
2. A mapping of this information to a form that is suitable for implementation in a directory that uses LDAP or X.500 as its access protocol

More information on directory-enabled networking can be obtained from the book *Directory Enabled Networks*, by John Strassner.

Object-Oriented Information Modeling

An information model is fundamentally different than a data model or a schema (Figure 52-1). Here are definitions of each:

- **Data model**—A concrete representation of the characteristics of a set of related objects in terms appropriate to a specific data storage and access technology
- **Schema**—A set of data models that describe a set of related objects to be managed
- **Information model**—A *technology-independent* specification of the characteristics of a set of objects, and their relationships to other objects in a managed environment, with no reference to storage methods, access protocols, or specific type of repositories

Figure 52-1 Information Models, Data Models, and Schemata

The primary purpose of the information model is to define a single universal representation of the data and objects to be managed that is independent of any specific storage technology and access protocol. The information model is used to define all appropriate objects in the environment that are to be managed and to show how they relate to each other.

Because the nature of the objects and the data describing these objects is different, it is therefore reasonable to expect that different data stores will be required to represent these objects and their interrelationships. For example, a policy might be written to change the type of queuing on a particular interface of an access router. This might be a function of the number of octets dropped and the number of users of specific service types (such as gold vs. silver vs. bronze service). Storing the results of an SNMP counter recording anything to do with the number of octets dropped is inappropriate for a directory because the counter data changes much too fast for the directory to keep up with. However, user service definitions, as well as the policy itself, are very appropriate to store in a directory because they can then take advantage of the replication mechanisms that directories have. As will be seen later in this chapter, directories are very well suited to serve as publication mechanisms; publishing data in a directory enables diverse applications to exchange and share data.

The advantage of the information model, then, is to be capable of representing how these different types of data and objects relate to each other in a single consistent manner without being biased by the capabilities of any one particular repository. Put another way, the information model specifies a *logical* repository that describes the objects and data to be managed. The logical repository maps into a set of *physical* data repositories. The specific set of data repositories to be used depends on the needs of the applications using the repositories. This enables the developer to choose the appropriate data store(s) and protocol(s) to use for a given application.

Applications have different needs, requiring different data stores. This isn't a problem—you simply build a set of mappings from the (single) information model to each type of data store that is being used. In general, these mappings will be different because each type of repository uses a specific type of storage technology that uses one or more particular access protocols. This makes one schema different

from another. For example, a directory schema is fundamentally different than a relational database schema. However, all schemata so derived can be related to each other because they are all derived from a universal information model.

Data Models Are Bound to Specific Types of Repositories

A data model represents the fundamental characteristics of an object or a set of objects *in a way that is specific to a particular type of repository*. For example, there are fundamental differences between a router object and a user object. Furthermore, each object will have a different implementation in a directory than in a relational database, even though the same information is represented in both schemata.

The directory implementation will consist of a set of entries that have attributes defined according to the syntaxes (such as the data types and ways that you can search for and find information in a directory) supported in LDAP and X.500. In addition, it emphasizes containment. *Containment* describes the subordinate relationships between one object and other objects in the system. In our example, a user object is usually “contained” in, or scoped by, a higher-level object, such as a group or an organizational unit (a fancy X.500 word for “division”).

The same user object implemented in a relational database will have a different structure than the same user object implemented in a directory. For example, data representing the user will be spread across one or more tables instead of existing within individual entries in a directory. Furthermore, the data will be structured slightly differently, to accommodate different data structures and access protocols that can be used in a database implementation compared to the directory implementation. Relationships to other objects, rather than containment of objects, is one of the main differences between a relational database implementation and a directory implementation.

An object-oriented information model uses object-oriented techniques to model information about a particular set of objects that exist in a managed environment. The key difference in an information model is that, in addition to describing the characteristics of entities, it also describes their behavior and interaction with each other. These latter two concepts may not be able to be captured in all repositories. Thus, the information model prescribes a means for relating different types of information, regardless of the type of data store that is being used. It is up to the developer to choose the right type of repository and other auxiliary tools to implement all facets of the information model if the repository itself is not capable of implementing the data and relationships in the information model.

An example may help to clarify this. Think of basing a decision to change the type of conditioning that a particular type of traffic is receiving on the network environment. This decision may depend on several factors:

- The number of dropped octets in a particular interface
- The service-level agreement assigned to a particular user or application
- Historical and other related information

These represent three fundamentally different types of information. Any one single data store is probably not optimal for storing this information because of the inherent differences in volume, frequency of update, types of queries, and data structures used to store and retrieve these data. The information model represents the relationships that each of these data structures have with each other and with other objects in the managed environment. This enables the developer to design optimized repositories to store each type of information and then recombine the data as appropriate. As another example, different data models could be used to model a router interface, users, and different types of services and application data that are provided on behalf of different users. However, the data model can't model the interaction between these objects. This is what the information model does. Therefore, we can see that different data models will be used to model different parts of the data described in the information model.

Thus, although directories are a very important type of repository for storing information about network elements and services, they are not the only type of data store that can be used. However, because directories usually contain the definitions of users, applications, and other network resources, they are often used in all applications to some extent. That is why this chapter concentrates on the mapping of DEN information to a form that enables DEN data to be stored and retrieved in a directory.

Realization of the Information Model

Currently, two important standard information models are being developed: the Common Information Model (CIM) and the Directory-Enabled Networks model, which is an extension of CIM. Both of these are currently governed by the DMTF.

The Common Information Model

CIM is an object-oriented information model that describes how a system and its components may be managed. It is defined by the Distributed Management Task Force (DMTF). Ongoing development of CIM is part of an industry-wide initiative for enabling enterprise management of devices and applications. A primary goal of CIM is the presentation of a consistent view of the managed environment, independent of the various protocols and data formats supported by those devices and applications. Many network infrastructure and management software providers have accepted CIM as an information model for enterprise management tools.

CIM is a layered information model, meaning that it consists of a set of submodels that build on and refine the knowledge present in outer, more generic layers. Specifically, a set of common abstractions and functions are defined in the core model (see Figure 52-2). These are then enhanced through the definition of submodels that are layered on, or use, the information in this core model. One of these layers is the network model, which came from DEN.

Figure 52-2 *The CIM Layered Information Model*



Version 2.2 of CIM consists of a core model, which is used to define concepts in the information model that apply to all areas of management. It is comprised of a set of classes, attributes, methods, and relationships that describe common concepts for managing systems and system components. The core model is the foundation for the class inheritance and relationship hierarchies, and is the basis for all common and extension models.

Common models are focused sets of classes, attributes, methods, and relationships that extend particular concepts in the core model. For example, the core model generically defines a service. The network model refines this concept to describe different types of services that are specific to networking, such as the forwarding and routing of traffic.

The best way to think of a common model is as a set of abstractions that frequently occur in a specific management domain. The seven common models are these:

- **System**—Defines key system components, such as computer system, operating system, file, and the relationships required to assemble them.
- **Device**—Defines how to realize physical devices in hardware and how to model connections between devices such as storage devices, media, sensors, printers, and power supplies.
- **Application**—Defines how to manage software installation within a system.
- **Network**—Defines refinement of the logical element class hierarchies to model network elements and services.
- **Physical**—Defines physical organization, containment structure, and compositions of devices and device interconnections.
- **User**—Models users, groups, and organizations, and shows how these objects interact with other components of a managed system.
- **Policy**—Builds on the original policy model proposed by DEN and provides a generic structure for representing and defining policy rules, conditions, and actions. It also specializes this to represent the specific requirement of QoS policy rules, conditions, and actions.

The combination of the core model and one or more common models provides the basis for a CIM- or DEN-compliant schema that can be bound to a specific application.

DEN, an Extension of CIM

DEN is two things:

- An extension of the information model defined in CIM that describes the physical and logical characteristics of network elements and services, as well as policies that control the provisioning and management of network elements and services
- A mapping of information to a format that can be stored in a directory that uses (L)DAP as its access protocol

The schemata for network integration defined in the DEN and CIM specifications are complementary. CIM is primarily concerned with the management of individual components in the context of an enterprise. DEN is primarily concerned with providing more detail about the networking components of a system, whether it is focused on the enterprise, the service provider, or both. This includes describing not just network elements and services, but also their provisioning and management through the use of policy objects.

The DEN schema, derived from the DEN information model, for mapping data in the DEN information model to a form suitable for implementation in a directory, incorporates concepts from both X.500 and CIM.

The utility of CIM is that it defines generic concepts of components to be managed in an environment. DEN extends CIM by adding information specific to networking that is more specialized than the information that CIM defines. The DEN mapping produces a directory schema that defines entries (along with other information) that can be added to an existing schema that represent network elements and services. It also defines entries that represent policy rules and related policy information.

The DEN information model and schema also incorporate information from the IETF Policy Framework working group (and possibly other working groups in the future) that has not yet been accepted by the DMTF.

A Brief Introduction to Directories

Today, the computing environment that must be managed includes not only the computers themselves, but also the network devices that connect them. Effective network management requires a variety of information from different sources, reflecting the different needs of the users of the network and the current state of the network. Furthermore, network management must be distributed throughout the various management points that are used to manage and control the network. Some of this information is appropriate for storing in directories, while other types are not. DEN prescribes a methodology to be used in modeling network elements and services so that information required for network provisioning and management may be implemented in whatever type of repository is appropriate. This usually involves directories, but it may also involve other types of repositories.

A directory service is a physically distributed, logically centralized repository of infrequently changing data that is used to manage the entire environment. Directories are commonly used to store information about users, applications, and network resources such as file servers and printers. DEN provides a schema that adds information to the directory. This schema in effect extends the directory, enabling it to contain information crucial for modeling network elements and services, as well as policies that control network elements and services. Better yet, DEN defines a schema that is independent of any particular directory vendor implementation.

Directories and Directory Services

This section provides a brief introduction to directories and directory services.

What Is a Directory?

A *directory* is used to record information about a particular group of objects. The directory is not intended to be a general-purpose data store. Rather, it is a special type of information repository whose primary purpose is to efficiently store and retrieve information about objects relevant to a particular application or set of applications.

Directory information is organized as shown in Figure 52-3. The groups of objects stored in a directory are organized in a hierarchical fashion. This is called the *directory information tree (DIT)*. The DIT consists of *directory objects*—each directory object corresponds to an entry in the DT. Each entry can have one or more attributes, and each attribute has at least one distinguished value (it may have more) and optionally additional nondistinguished values. This structure enables you to retrieve information either by specifying an exact set of criteria to be matched, or by specifying a more general set of criteria that describes the characteristics of the information that you are seeking.

Figure 52-3 *The Structure of Directory Information*



Distinguished values are used to compute relative distinguished names (RDNs) and fully qualified distinguished names (FQDNs). The FQDN for an entry is built by taking the FQDN of its parent entry and appending the RDN specified in the entry. Thus, the FQDN at any level is the set of RDNs that together specify a path from the root of the DIT to that particular directory entry. This is shown in Figure 52-4.

Figure 52-4 Directory Entries, RDNs, and FQDNs

An object in the real world can be modeled as one or more directory entries. Each directory entry is an object that has a set of characteristics describing the information carried by the object. These characteristics are implemented as attributes of the entry. For example, a User object might have attributes that define the first name, last name, employee ID, phone number, and other data associated with that user. Each of these attributes is common to all instances of the User class; however, the specific values of at least some of these attributes will be different so that different users can be identified.

The set of attributes that an entry has is determined by the *object class* of that entry. The object class defines which attributes must be included (for example, values must be specified for them) and which attributes may be included for a given entry (for example, they are defined in the schema but do not have to be instantiated). The complete set of object classes and attributes for a directory is defined as the *schema* of the directory.

Each attribute has a specific data type that may have restrictions qualifying the values of that data type (such as a string with alphanumeric characters only). This is called the *syntax* of the attribute. In addition, a predefined set of matching rules is defined for each entry to specify whether this attribute is considered a match for a search. (For example, given a string, ignore the case of each character in the string and see whether this attribute's value equals the value that is being searched for.) Attributes may also be multivalued, but they do not have to be. Finally, all attributes have object identifiers that uniquely define them. These are *ASN.1 identifiers*.

Characteristics of a Directory

Directories have five important characteristics:

- The storage of information is optimized so that it can be read much more frequently than it is written.
- Information is stored in a *hierarchical* fashion.
- Information in a directory is *attribute-based*.
- Directories provide a *unified namespace* for all resources for which they contain information.
- Directories can efficiently distribute information in a distributed system through replication.

The first point means that directories are very good at performing high-volume search operations (such as searching an address book), but not good at performing operations that require frequent writing (such as navigating an airline reservation system).

The second and third points are somewhat related. The second point means that the information infrastructure is based on parent-child relationships. Containment, not inheritance, is the driving factor of a good directory design. The third point refers to the fact that the directory is comprised of a set of objects. Each of the objects has a set of attributes that contain the information. Thus, the information is spread through the attributes of the objects that form the infrastructure of the directory.

The fourth point is very important. It means that common information can be located and shared by different directory clients because each application can use the same method of referencing an object. A unified namespace enables network elements and services to be seamlessly integrated with other types of information, such as users, applications, and servers.

The final point is critical for building an information infrastructure. The directory server has the capability to control what information gets distributed when and to what other nodes in the system.

What Is a Directory Service?

A *directory service* stores and retrieves information from the directory on behalf of one or more authorized users.

Directory services are built to provide certain types of application-specific information. However, multiple directory services can share the same directory. For example, think of two telephone books, a White Pages phone book and a Yellow Pages phone book. Both provide phone numbers, but in a different way. The White Pages phone book enables you to find the telephone number of a person, while the Yellow Pages phone book enables you to look up categories of information and retrieve multiple phone numbers.

A directory can be used to implement both of these services. In this example, the directory would contain the data model describing the different types of users and services that you are interested in. There are actually two different directory services, one for providing access to White Pages data and one for Yellow Pages data. However, they can use the same directory—the data model of the White Pages service is simply extended to suit the more complex needs of the Yellow Pages service.

This example also shows that directory services are usually restricted to operate in a particular way. For example, you can't give the telephone number of a user to a White Pages service and (easily) get the corresponding user.

Current Uses of the Directory

A traditional directory service provides a means for locating and identifying users and available resources in a distributed system. Directory services also provide the foundation for adding, modifying, removing, renaming, and managing system components without disrupting the services provided by other system components. Today's directory services are used to do the following:

- Store information about system components in a distributed manner. The directory is replicated among several servers so that a user or service needing access to the directory can query a local server for the information.
- Support common searching needs, such as by attribute (for example, “Find the phone number for James Smith”) and by classification (for example, “Find all color printers on the third floor”).
- Provide important information to enable single-user logon to services, resources, and applications.
- Enable a location-independent point of administration and management. Note that administrative tools do not have to be centrally located and managed.
- Replicate data to provide consistent access. Modifications made to any replica of the directory are propagated around the network so that any application accessing the directory anywhere sees consistent information after the change is propagated.

Motivation for DEN and Intelligent Networking

There are two major problems with pre-DEN directory servers and services that have prevented them from being used for intelligent networking. The first is the incapability for heterogeneous directory servers to replicate data with each other. The problem is that the LDAP protocol does not provide for this, and directory vendors have implemented their own proprietary replication mechanisms. Some directory service vendors use a form of synchronization, which is a tool that can read information from another vendor’s directory server and translate that information into a format that can be used by the server performing the synchronization.

However, there are no synchronization standards (each works differently), and there are a variety of limitations with each implementation. It should be noted that this problem is currently being worked on in the IETF, in the LDAP Duplication and Update Protocol working group (LDUP). LDUP has defined an information model that has been used to guide the development of a replication protocol that is in the process of being standardized. This work should hopefully be finished by the end of 2000. More information about LDUP can be obtained at <http://www.ietf.org/html.charters/ldup-charter.html>.

The second problem is the lack of standardization in representing information. For example, there are many competing ways for representing generic user information. In addition, there is a tendency to design applications in a stovepipe fashion. This means that applications tend to represent data according to a set of naming conventions and structures that makes most sense to their use. Although this is beneficial for a single application, it makes it hard for different applications to share and reuse data. This problem is illustrated in Figure 52-5.

Figure 52-5 Integration of Stovepipe Applications Is Very Difficult

As can be seen in Figure 52-5, the proliferation of disparate data stores, each built to support a particular application's needs, makes integration very difficult. The first problem is the continuing use of application-specific repositories. This is because each repository will define some of the same data using different storage and naming rules. This causes synchronization problems because now each copy of the data must be updated at the same time; because they are in different formats and representations, however, this is quite hard. Second, it results in different views of the same data. The final issue is integration. If all applications are using private versions of the same data model, how will they exchange data? Note that this also precludes the capability of applications to share and reuse the others' data.

Although this was a problem, a worse problem was that there was no standard at all for representing network elements and services before DEN. Therefore, DEN provides two important benefits:

- Network elements and services are represented in a standard way, enabling diverse applications to share and reuse the same data.
- All objects of a managed environment are represented as objects. This enables the different types of entities that make up a managed system to be treated in the same way. This provides a unified way of representing information about different types of entities.

As an example, think of the current way of providing unified network management. You might use HP/Openview for managing the different entities in an enterprise, and one or more Cisco-specific applications (assume for the sake of argument that they are running on Windows NT) for configuring and provisioning the Cisco devices in the enterprise network. This is a problem because these two types of applications need to share data. But this is very hard, if not impossible, because of these reasons:

- The applications have different ways of representing the same information (because they are built differently).
- The applications are running on different platforms.
- The applications are coded in different languages.
- Different user interfaces exist for each application.

Note that, in general, writing APIs doesn't work. This is not only because of the previously stated reasons, but also because a given API is usually a reflection of the internal functionality of the application. This requires the developer to have access to and be familiar with the operation of the application to be integrated. This clearly is not the usual case—and even if it were, each time that the applications being integrated change, the APIs would have to change.

DEN solves this by defining a standard way to represent information. By using techniques such as XML, developers can encode their data as represented in DEN and can ship it to another application on a different platform. That application can then decode the DEN data and use it directly in its own interface. Clearly, this is a very powerful concept:

- The administrator needs to learn only one application.
- APIs don't have to be built only to break with each change of each application.
- Data can be reused and shared between applications, which enables best-of-breed applications to work together seamlessly.

DEN therefore enables different vendors to build different network elements and applications that can communicate with each other. This enables various types of systems as well as network elements to be equal partners in implementing and reacting to decisions made throughout the networked environment.

Distributing Intelligence in Networked Applications

Rapid Internet growth over the past several years has created the need for more robust, scalable, and secure networking services. Residential customers desire rich multimedia services, such as data and video. Corporate customers are looking to telcos and service providers for powerful yet affordable services. Users want a reliable, easy-to-use, friendly service.

A fundamental shift toward bandwidth-intensive and isochronous network applications has occurred. Communication problems are no longer just a function of bandwidth. Rather, it is increasingly more important to understand the needs of different types of traffic flowing in the network and to design a network that can accommodate those needs. Furthermore, if resources become scarce, then an efficient way to allocate these resources according to the business rules of the company is required.

DEN plays a critical role in solving both of these problems. The information model is used to describe the function and needs of the different applications using the network. This translates into a set of traffic flows that will use the network. The DEN policy model can be used to translate from business terms to a form that is independent of any one particular device. This can then be used to map to device-specific protocols and mechanisms.

DEN Policy Model

The desire to allocate resources according to the business rules of the company is a critical requirement. The DEN policy model defined a continuum of policies, each optimized to represent different sets of information. For example, a business goal might be administrator-defined, which makes it device- and mechanism-independent. As an example, consider this business rule:

```
IF
  User is subscribed to gold service,
THEN
  allow use of NetMeeting and
  provide premium data services
ENDIF
```

This is a perfectly valid business rule, but it doesn't say how to configure the devices. It does, however, say what services should be allocated.

This business rule needs to be translated to device configuration rules so that the network can support the business policies of the organization. One such translation might be this:

```
IF
  SourceIPAddress = 172.3.128.0/15
THEN
```

```
Mark Voice with EF and
Mark Data with AF11
ENDIF
```

This rule starts mapping the services specified in the business rule to a form that can be applied to a device. This rule is device-independent in that it can apply to many different types of devices.

The next step is to map this to a form that can be implemented in a device. This means that we need to map the previous rule to a form that identifies the device mechanisms that must be controlled. There are several different forms of this, each appropriate for different actions:

- Configure component so that it can be used to condition forwarded traffic
- Configure component so that it can act on traffic directly
- Trigger action based on a network or system event (such as link failure)

This set of policy rules can now be translated into (for example) a set of device-specific CLI commands.

The advantage of this approach is that it can be used as a reusable template. That is, instead of trying to perform these mappings for each interface of each device in the network, a set of templates controlled by policy can be developed so that the device-independent rules can be separated from the device-dependent rules. The Policy Framework working group of the IETF is taking exactly this approach for the control and provisioning of QoS; see <http://www.ietf.org/html.charters/policy-charter.html> for more information.

Use of the Directory in Intelligent Networking

A directory service can be used to store and retrieve much of this information. This is because of the following four main reasons:

1. A directory is a natural publishing medium, capable of supporting a high number of reads as well as allowing arbitrary information to be stored and retrieved. Thus, there are no restrictions on the information itself; this provides inherent extensibility for accommodating additional as well as new information.
2. Directories are the *de facto* standard for containing user information and other types of information, and directory-enabled network applications require user, network, and other types of resource information to be integrated. The advantage is that information about network resources, elements, and services are not only colocated, but they are represented as equal objects that have a common representation. This enables the different applications that want to use and share this information to access a single repository. This greatly simplifies the design of the overall system.
3. Directories facilitate finding information without knowing the complete path or name of the object that has that information. A directory service is more than a naming service, such as DNS. A directory service enables both the searching and the retrieval of named information.
4. A directory can also be used to point to other systems that contain information; this provides a single place where applications can go to find information.

Challenges of Current Directory Services

Current directory services technology is not designed to meet the ever-increasing demands of today's public and private network applications. This is because current directory services were built mainly to accommodate administrative needs. Directories used in

this fashion take the form of dumb warehouses, where they are simply used to store simple information. The directory must be transformed from a dumb warehouse to an authoritative, distributed, intelligent repository of information for services and applications. Viewed in this way, the directory is one of the foundations for an intelligent infrastructure.

Bandwidth-intensive and isochronous network applications require that the devices that lie on a path through the network between source and end devices be configured appropriately if they are to function properly. This configuration is often dynamic, taking place on demand when a particular user logs on to the network from any of a number of possible locations. Only when management information about the users, network devices, and services involved is available in a single, authoritative location is it possible to actually manage this new class of applications.

An Overview of DEN

This section defines the problem domains, information model, and usage for integrating networks with directory services. Directory-enabled networking is a design philosophy that uses the DEN specification to model components in a managed environment. These components include network devices, host systems, operating systems, management tools, and other components of a system to be managed. All these components use the directory service to do the following:

- Publish information about themselves
- Discover other resources
- Obtain information about other resources

DEN is two things:

1. An extension of CIM
2. A mapping of information to a format that can be stored in a directory that uses (L)DAP as its access protocol

The following sections provide an overview of building interoperable network-enabled solutions and the benefits that DEN can bring.

Networks and DEN

Administrative needs and the tools that service them have evolved as distributed systems have evolved. Today's directory services were designed to provide central management of security and contact information in a network with a relatively small number of relatively large computers. Network management has been the province of more specialized tools, each with its own information store. Application management has been addressed as an afterthought when it has been addressed at all.

Obtaining convergence on the structure and representation of information in any one type of repository (let alone across all the different information stores that are applicable to networking) has been very difficult. The result is an environment in which vertical management tools have proliferated. Lack of integration and the sheer complexity of the tools themselves has become a barrier to the deployment of new applications.

Administrators need a level of control over their networks that is currently unavailable. Streaming multimedia, use of public networks and the attendant security concerns, and rapidly growing user communities present a tremendous challenge.

Simply managing individual devices is no longer sufficient. Network administrators need to define and manage policies to control the network and its resources in a distributed yet logically centralized manner. In general terms, policies define what resources a given consumer can use in the context of a given application or service. The incapability to easily manage policies is a significant barrier to deployment of leading-edge distributed applications.

A consumer is a user, an application, a service, or another user of resources.

Defining and managing policies requires a common store of well-defined information about the network and its resources—users, applications, devices, protocols, and media—and the relationships among these elements. This is information about the network as well as the information traditionally viewed as defining the network (for example, routing tables).

At issue is where to store policy and other information that needs to be applied across components in a way that makes it usable by a broad range of consumers.

A scalable, secure directory service that presents a logically centralized view of physically distributed information is the logical place to store the metainformation essential to creating and managing a next-generation network. The specification for the integration of directory services and network services defines the information model and schema to make this possible.

Two of the promises of DEN are these:

- To define a means of storing data in a common repository
- To provide a way for applications to be capable of taking advantage of data managed by other applications.

This represents a fundamentally new way of thinking about network management applications, along with applications that seek to leverage the power of the network. One example of this is to compare traditional network management with network management that uses DEN. In a traditional network management system, each device in the network is represented once. However, each device has detailed configuration information that is stored not in the network management system, but in either the application itself or another data store. The role of portraying the device in the network management system is to enable the user to launch a particular management application that is focused on one or more aspects of managing that device. Thus, the network management system provides a common place to represent the device, but not to store its information. This makes integrating applications and sharing information between different applications difficult, if not impossible.

This is very different than a directory-based approach that uses DEN. The fundamental purpose of DEN is to provide a common, unifying information repository that is used to store data and information about the data (such as metadata) for multiple applications to share and use. For example, consider a network management system using HP/Openview running on HP/UX. Suppose that it discovers a new router, one that it doesn't have any information on in its internal database. Before DEN, the only solution for the network administrator would be to purchase another management tool that supports the new router. Then, every time that router must be managed, the network administrator would have to get up and change consoles. Of course, this is only the beginning of the problems because the new management tool probably has its user interface and runs on a different platform.

With DEN, things have the potential to be seamless. I'll use an example from the 1999 N+I show in Atlanta, where HP and Cisco demonstrated this. Both HP and Cisco support DEN, which enables a set of common management information defined by the CIM and DEN standards to be exchanged (note that more detailed information can be exchanged by subclassing these standards and using them as the basis to represent Cisco-specific products and services). This was done by having the HP/Openview agent message the Cisco management agent, asking for the DEN description of the device. This information was encoded in XML and was shipped to HP/Openview over HTTP. The combination of XML and HTTP ensured that no platform- and language-specific problems got in the way.

However, the real bonus is that *the native Cisco router data is used to populate the HP/Openview screen*. This has the following implications:

- The administrator has to learn only one user interface.
- The system is inherently extensible. Because DEN is their common interface, it can dynamically accommodate new products as long as they are described in DEN.
- No complicated APIs must be built.
- Additional products that want to share data can do so.

Directory Service and Network Management

Network elements typically have a dynamic state and a persistent state. Dynamic state is well addressed by network management protocols. However, there is no standard way to describe and store persistent state. Moreover, existing tools and applications focus on managing individual network elements rather than the entire network. The DEN and CIM specifications define a standard schema for storing persistent state and an information model for describing the relationships among objects representing users, applications, network elements, and network services (see Figure 52-6). Network-management protocols (such as SNMP, CMIP, and HMMP) are used to talk to the network elements.

The network schema extensions for the directory service are used to talk about network elements and services.

The integration of the network infrastructure with the directory service allows the applications and users to discover the existence of devices and relationships by querying the directory service rather than contacting the individual devices and aggregating the results. Exposing network elements in the directory enhances manageability and usability while reducing the load on the network. The end-user and administrator experience is enhanced because there is a single, authoritative place to obtain the information of interest.

The Extended Schema and Other Device Schemata

Schemata defined by SNMP (MIBs), DMTF CIM, and so on, are intended primarily to address the details of individual devices. The intent of the integrated, extended schema is to leverage the information exposed by existing schemata and management frameworks, not to replace them. Furthermore, the CIM and DEN information models are repository-independent. This means that the devices that are to be represented in and managed by a DEN schema do not themselves have to implement LDAP.

Figure 52-6 Sampling of Important Base Classes of the DEN Schema



Network Applications Integrated with the Directory and Other Network Protocols

The schema and information model defined augments existing network services and associated protocols, such as Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), and RADIUS.

The directory provides a common store for network information; the information model describes the relationships that can be represented in the directory. The usage model defines how existing network services and protocols work with the elements in the information model to accomplish specific goals, such as coordinating IP address allocation across multiple DHCP servers, establishing and propagating remote access login policy, and so on.

Benefits of Using DEN

DEN had three main use cases. The first was to help simplify device configuration. Device configuration has recently become increasingly complex, mainly because of two important factors. First, different types of users and applications are vying for limited network resources at the same time. The problem is *not* lack of bandwidth, but rather *traffic mix* (that is, how these different applications, all with their own specific needs, peacefully coexist in the same network). This has caused network device vendors to add more functionality in their devices. Thus, network devices are asked to do more, resulting in increasingly complex device configurations.

The second use case was to control the management and provisioning of network devices through the use of policies. The business community wanted a way to map service-level agreements and business rules to a common set of policies. These policies would *control the allocation of network resources* based on user, subnet, time-of-day, or other appropriate factors. Most importantly, they ensure that services are implemented in a hardware-independent way. Of course, this cannot be done without a standard information model.

The third use case is to define a means to make applications more network-aware and to make the network more application-aware. This is accomplished in the DEN information model by ensuring that network elements, services, and other components of a managed environment are all represented using objects. If all objects are equal and have equal capabilities, then they can all be represented equally well, and communication between them is assured.

Directory-Enabled Networks services benefit different constituencies in different ways.

For the end user, it helps enable single sign-on services. Single sign-on services enable the same set of access rights and privileges to be provided no matter where, when, or how the user logs on to the network (within the limits set by the policies of the system, of course). In addition, it enables individuals to be identified and provided services proportionate to their role in the company, service contract, and so on. It also helps companies enforce sophisticated policies. For example, a business rule may prohibit shipping code over the public Internet. Thus, even though a user successfully authenticates over a dialup line, the policy will correctly deny authorization to connect to a code server because the system recognizes that the user is connecting over the public Internet. The key technology used here is DEN's robust notions of services and policies, and the capability to link them to users as well as devices.

Service providers are interested in directory-enabled networking because they need a way to provide differentiated services. The \$19.95 "all you can eat" philosophy doesn't even cover the cost of building out new networking infrastructure. In addition, they are interested in using directory-enabled networking to facilitate turning on new services through central management. The key advantage used here is the capability to define policy-based management of network elements and services, and to isolate the effect of turning on new services to a portion of the network.

Enterprise customers need a centralized way to protect mission-critical traffic and to better manage increasingly complex device configurations. DEN's capability to associate multiple traffic streams with a single application, along with its capability to define policy-based management of network elements and services, is critical here.

Application developers are provided a standard means of representing information describing network elements and services. This enables them to better leverage the power of the network. DEN's capability to describe applications, traffic that they generate, and how to manage these sets of traffic through policy is essential to implementing this goal.

How DEN Is Used in Cisco Products

Figure 52-7 shows the two layers of mapping that are inherent in DEN. The first is from the information model to a target repository. This mapping defines the type of repository to be used, which in turn defines the way data is stored, the set of data structures that can be used, the protocol(s) that will be used to store and retrieve the data, and other factors. The second mapping is sometimes required either to optimize the implementation to suit some application-specific needs, or because different vendors do not implement the same features in the same type of repository.

Figure 52-7 DEN Mappings

Cisco is using this philosophy to standardize its use of DEN. In fact, Cisco is building three models. The first is a standard mapping of the DEN information model into a set of implementations. A directory implementation of CIM 2.2 and the policy model will ship early in the second quarter of 2000.

The second is a set of common Cisco-specific extensions being developed by a cross-product group of Cisco engineers. This model extends the generic concepts of CIM and DEN to a device-independent intermediate layer. For example, the concept of a port is enhanced and linked to Cisco-specific network elements and services.

The third is a set of application-specific extensions that model Cisco devices and services. This set of models is derived from DEN and the Cisco extensions to DEN. It enables specific Cisco devices and services to be explicitly modeled to a very fine level of detail in the information model.

It should be noted that by basing the Cisco-specific extensions on DEN, Cisco proprietary network elements and services are modeled based on a standard. This is much better and more powerful than if Cisco had decided to base its work on either a competing standard or (worse) no standard at all. It guarantees a level of interoperability with the standard, and it opens the way in the future to exchange more detailed information with its partners. The same is true for the application-specific extensions that are based on the Cisco extensions.

The Directory-Enabled Networking Vision

The vision for enhancing networking through integration with the directory service is to provide network-enabled applications appropriate information from the directory. Eventually, intelligent network applications will transparently leverage the network on behalf of the user. The development of intelligent networks can be achieved through the following steps:

- Relying on a robust directory service
- Adding a standards-based schema for modeling network elements and services
- Adding protocols for accessing, managing, and manipulating directory information

The goals of work in developing directory-enabled networks are listed here:

- To provide support for applications that have the capability to leverage the network infrastructure transparently on behalf of the end user
- To provide a robust, extensible foundation for building network-centric applications
- To enable end-to-end network services on a per-user basis
- To enable network-wide service creation and provisioning
- To enable network-wide management

The focus is on providing management of the network as a system, not a set of disparate components or individual device interfaces. Using directory services to define the relationship among components allows the network manager to manage the network as a system. Vendors have adopted *de facto* open industry standards, such as DNS and DHCP, to tie these services into their enterprise management systems. DEN is the next such standard.

Summary

This chapter has provided a brief introduction to Directory-Enabled Networks. DEN is two very important things. First and foremost, it is an object-oriented information model that is used to describe entities to be managed in an environment. Although there are many such models, DEN is unique in that it is the only model to describe, in a repository-independent fashion, both networking elements and services as well as other objects that together constitute a managed environment. Second, DEN defines a mapping for the data specified in the DEN information model to a form that can be stored and retrieved in a directory (which uses either LDAP or X.500 as its access protocol).

A brief introduction to object-oriented information modeling, and the benefits of using such an approach, was described. This method enables any entity that needs to be managed to be modeled in a consistent manner in the managed environment. Directories are one important example of mapping this information. This is because directories already contain important information, such as users, printers, and other network resources. Conceptually, DEN extends the type of data that can be modeled in directories, and shows how that information is related to different types of data in other types of data stores.

DEN forms a cornerstone of building intelligent network services, as well as controlling systems through policies. DEN models the network as a provider of intelligent services, and models clients of the network as users of those services. This provides a methodology to make applications more aware of the network and to make the network more aware of the needs of various applications.

Finally, examples of how DEN is used within Cisco Systems to build a new set of intelligent products and solutions were provided.

Review Questions

Q—*What is DEN?*

A—DEN stands for Directory-Enabled Networks, a specification that defines different entities in a managed system using an object-oriented information model that is independent of repository and access protocol. DEN also defines a mapping of the data in the information model to a form that can be stored and retrieved from a directory that uses (L)DAP as its access protocol

Q—*Does DEN require the use of a directory?*

A—No. DEN is, first and foremost, an object-oriented information model that is *independent* of repository and access protocol. Data can be mapped to a directory, but also to other types of data stores (such as a relational or object database).

Q—*Is DEN just about modeling network devices and services?*

A—No. Although DEN concentrates on building a robust and extensible infrastructure that can model different network elements and services, one of its primary benefits is that it treats all types of entities in the managed environment as equal objects.

Q—*What is an object-oriented information model?*

A—An object-oriented information model is a means of using object-oriented techniques to design a set of classes and relationships to represent the different objects in a managed environment.

Q—*Name some of the important benefits of DEN.*

A—First and foremost, DEN is an object-oriented information model that describes different components of a managed environment in a common way. This enables a close relationship to be established between classes that define network elements, and services and classes that define other objects. This is the primary mechanism used to define which network services a client needs.

Second, DEN is object-oriented, so it is inherently extensible. This means that concepts not yet defined in DEN can be easily modeled and added to the DEN standard.

Third, DEN enables the application developer as well as the network designer to think of the network as a provider of intelligent services. This enables application developers to describe the functions and treatment that the traffic of their applications requires in terms that the network can represent directly. Thus, if a certain application has specific jitter and latency requirements, DEN can be used to define the set of services that together meet these requirements.

Fourth, and closely related, DEN enables businesses to prioritize the treatment of different applications that are vying for network resources. This enables a business administrator to write a policy that says that SAP and PeopleSoft applications should get preferential treatment over FTP traffic. This enables the network to be designed to treat the applications that a business runs according to the business rules of that organization.

A final example benefit of DEN (although there are more) is that DEN is a standard. This means that it can be used by network vendors, system integrators, and others to define a common framework to describe, define, share, and reuse data.

Q—*How does DEN model relationships between objects?*

A—This is one of the crucial advantages of the DEN approach. DEN is not just a set of data models describing the characteristics of managed objects. DEN also defines a set of relationships between these objects. Without such a set of relationships, you could not relate the specific set of services that must be used to provision different applications for different users. In addition, DEN implements these relationships as classes. This enables all the benefits of object-orientation (such as subclassing, putting properties and methods on the relationship itself, and so on) to be applied to the relationship. Note that DEN is unique in this respect among the different modeling approaches that exist.

For More Information

DEN and Related Standards Work

The Desktop Management Task Force (DMTF) is the industry consortium chartered with development, support, and maintenance of management standards for PC systems and products, including CIM and DEN. More information can be obtained from <http://www.dmtf.org>.

Working Groups in the IETF

The charter of the Policy Framework Working Group of the IETF is available from <http://www.ietf.org/html.charters/policy-charter.html>.

The LDAPEXT (LDAP Extensions) Working Group of the IETF is chartered with continuing to develop an Internet directory service. The LDAPEXT Working Group defines and standardizes extensions to the LDAP Version 3 protocol, extensions to the use of LDAP on the Internet, and the API to LDAP. More information can be obtained from <http://www.ietf.org/html.charters/ldapext-charter.html>.

The LDUP (LDAP Duplication and Update Protocol) Working Group of the IETF is chartered with defining additions (protocol and schemata) to the LDAP protocol to enable different directory vendors to replicate with each other. More information can be obtained from <http://www.ietf.org/html.charters/ldup-charter.html>.

The RAP (RSVP Admission Policy) Working Group of the IETF is concerned with developing standards for enabling a scalable policy control model that can provide quality of service on the Internet using explicit signaling protocols such as RSVP. Common Open Policy Service (COPS) defines a protocol to transmit policy requests and responses. More information on both can be found at <http://www.ietf.org/html.charters/rap-charter.html>.

Simple Network Management Protocol (SNMP) is the de facto management standard for IP-based systems. Several IETF working groups actively work on the development of SNMP. Two to examine are these:

- <http://www.ietf.org/html.charters/agentx-charter.html>
- <http://www.ietf.org/html.charters/snmpv3-charter.html>

The goal of the first working group is to make the SNMP Agent more extensible. The goal of the SNMPv3 Working Group is to define the next generation of SNMP.

The Remote Network Monitoring Management Information Base is available in two versions. The following RFCs define it:

- <http://info.internet.isi.edu/in-notes/rfc/files/rfc1757.txt>
("Remote Network Monitoring Management Information Base")
- <http://info.internet.isi.edu/in-notes/rfc/files/rfc2021.txt>
("Remote Network Monitoring Management Information Base v2 Using SMI v2")
- <http://info.internet.isi.edu/in-notes/rfc/files/rfc2074.txt>
("Remote Network Monitoring MIB Protocol Identifiers")

RSVP is defined by several RFCs. The ones most relevant to this book are listed here. Also check the RAP Working Group. Go to the URL <http://info.internet.isi.edu/in-notes/rfc/files/>. Then pull the following files:

- [rfc2205.txt](#)—"RSVP Functional Specification"
- [rfc2206.txt](#)—"RSVP Management Information Base Using SMIv2"
- [rfc2207.txt](#)—"RSVP Extensions for IPSec Data Flows"
- [rfc2208.txt](#)—"RSVP Applicability Statement"
- [rfc2209.txt](#)—"RSVP Message Processing Rules"
- [rfc2210.txt](#)—"The Use of RSVP with IETF Integrated Services"

The Differentiated Services IETF Working Group is defining "relatively simple and coarse methods of providing differentiated classes of service for Internet traffic." Specifically, a small set of building blocks is defined that enables quality of service to be defined on a per-hop basis. This work is described in <http://www.ietf.org/html.charters/diffserv-charter.html>.

The Integrated Services Working Group of the IETF Recent Experiments demonstrates the capability of packet-switching protocols to support integrated services—the transport of audio, video, real-time, and classical data traffic within a single network infrastructure. More information can be obtained at <http://www.ietf.org/html.charters/intserv-charter.html>.

References on Directories

Probably the best source of information on directories is in the two IETF working groups LDAPEXT and LDUP, which were described previously. Two additional public URLs that contain some great information on directories are these:

- <http://www.critical-angle.com/ldapworld/>
- <http://www.kingsmountain.com/ldapRoadmap.shtml>

Strassner, John. *Directory Enabled Networks*. Indianapolis: Macmillan Technical Publishing, 1999.

■ For More Information